Codeword in binary serial form or parallel 16-bit form

100

```
d0d1d2d3d4d5d6d7 e0e1e2e3e4e5e6e7
  [Di]= st          [Ei]= uv
```

Figure 1A

Codeword in binary byte form

110

```
d0d1d2d3d4d5d6d7  [Di]= st
e0e1e2e3e4e5e6e7  [Ei]= uv
```

Figure 1B

Codeword in binary array form showing row and column erasure elements

120

```
[Di]   d0d1d2d3 s   {hexadecimal row elements}
       d4d5d6d7 t
[Ei]   e0e1e2e3 u
       e4e5e6e7 v
       ---------
        w x y z     {hexadecimal column elements}
```

Figure 1C

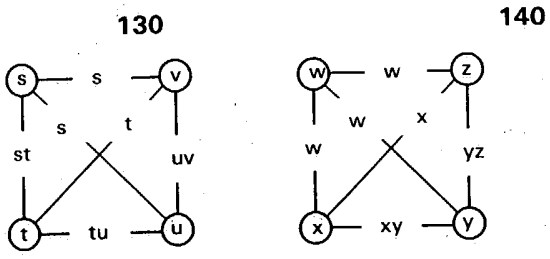
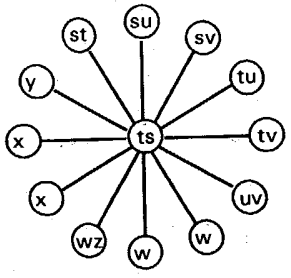


Figure 1D



150

micro-mirrors
12 per codeword

Figure 1E

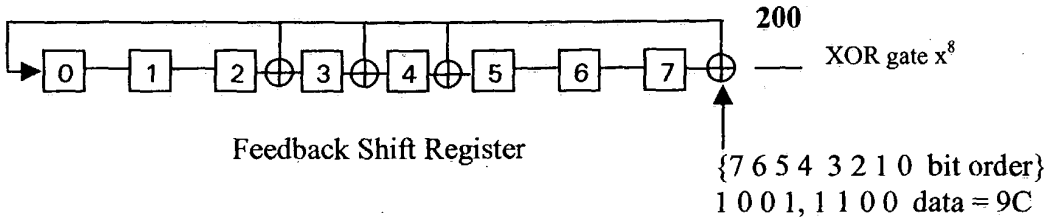


Figure 2A

0	0	0	0	0	0	0	0	
1	0	0	1	1	1	0	0	results after 1 shift
0	1	0	0	1	1	1	0	results after 2 shifts
0	0	1	0	0	1	1	1	results after 3 shifts
0	0	0	1	0	0	1	1	results after 4 shifts
0	0	0	0	1	0	0	1	results after 5 shifts
0	0	0	0	0	1	0	0	results after 6 shifts
0	0	0	0	0	0	1	0	results after 7 shifts
0	0	0	0	0	0	0	1	results after 8 shifts, ECC = 80

Figure 2B

210

$e0 = d0 + d3 + d4 + d5 + d6$
$e1 = d1 + d4 + d5 + d6 + d7$
$e2 = d2 + d5 + d6 + d7$
$e3 = d0 + d4 + d5 + d7$
$e4 = d0 + d1 + d3 + d4$
$e5 = d0 + d1 + d2 + d3 + d6$
$e6 = d1 + d2 + d3 + d4 + d7$
$e7 = d2 + d3 + d4 + d5$

per example
$0 + 1 + 1 + 0 + 0 = 0$
$0 + 1 + 0 + 0 + 1 = 0$
$1 + 0 + 0 + 1 = 0$
$0 + 1 + 0 + 1 = 0$
$0 + 0 + 1 + 1 = 0$
$0 + 0 + 1 + 1 + 0 = 0$
$0 + 1 + 1 + 1 + 1 = 0$
$1 + 1 + 1 + 0 = 1$
ECC = 80 for data byte 9C

Figure 2C

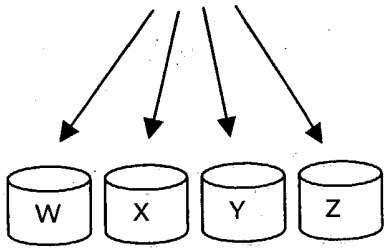
220

$d0 = e2 + e3 + e4 + e5$
$d1 = e0 + e3 + e4 + e5 + e6$
$d2 = e1 + e4 + e5 + e6 + e7$
$d3 = e3 + e4 + e6 + e7$
$d4 = e0 + e2 + e3 + e7$
$d5 = e0 + e1 + e2 + e5$
$d6 = e0 + e1 + e2 + e3 + e6$
$d7 = e1 + e2 + e3 + e4 + e7$

per example
$0 + 0 + 0 + 0 = 0$
$0 + 0 + 0 + 0 + 0 = 0$
$0 + 0 + 0 + 0 + 1 = 1$
$0 + 0 + 0 + 1 = 1$
$0 + 0 + 0 + 1 = 1$
$0 + 0 + 0 + 0 = 0$
$0 + 0 + 0 + 0 + 0 = 0$
$0 + 0 + 0 + 0 + 1 = 1$
data byte = 9C for ECC = 80

Figure 2D

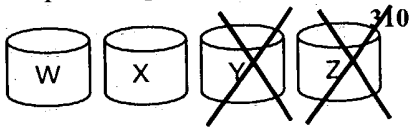
		w x y z	0 1 2 3 data bit position
data byte 3	1 F	7 1 5 9	1 1 1 1 [D] 1 0 0 0 1 0 1 0 [E] ECC = 85 0 0 0 1
data byte 2	D2	2 1 A A	0 1 0 0 [D] 1 0 1 1 0 0 0 0 [E] ECC = C0 0 0 1 1
data byte 1	4 7	1 1 3 C	1 1 1 0 [D] 0 0 1 0 0 0 0 1 [E] ECC = 88 0 0 0 1



disk array WXYZ

dispersed codewords

W	X	Y	Z
7	1	5	9
2	1	A	A
1	1	3	C



both Y and Z drives fail
read array WX

W	X
7	1
2	1
1	1

Recovered Data

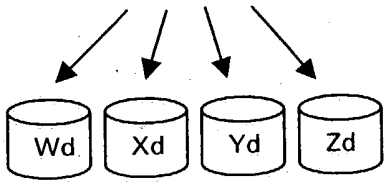
1F
D2
47

Figure 3

	data	w x y z	0 1 2 3 data bit position
data byte 2	D2	0 1 0 0 1 0 1 1	0 1 0 0 [D] 1 0 1 1 0 0 0 0 [E] note – ECC = C0 0 0 1 1
data byte 1	4 7	1 1 1 0 0 0 1 0	1 1 1 0 [D] 0 0 1 0 0 0 0 1 [E] note – ECC = 88 0 0 0 1
		6 D 7 4	

write array WXYZ

primary data
data array WXYZ



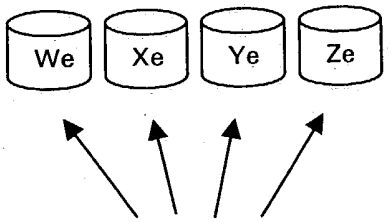
400

dispersed codewords

2	1	A	A
1	1	3	C

mirrored "proxy" data
ECC array WXYZ

410



0 0 4 7

ECC byte 2	C0	0 0 0 0 0 0 1 1	C0 / D2	see Table 2, field 7
ECC byte 1	88	0 0 0 1 0 0 0 1	88 / 47	

Figure 4

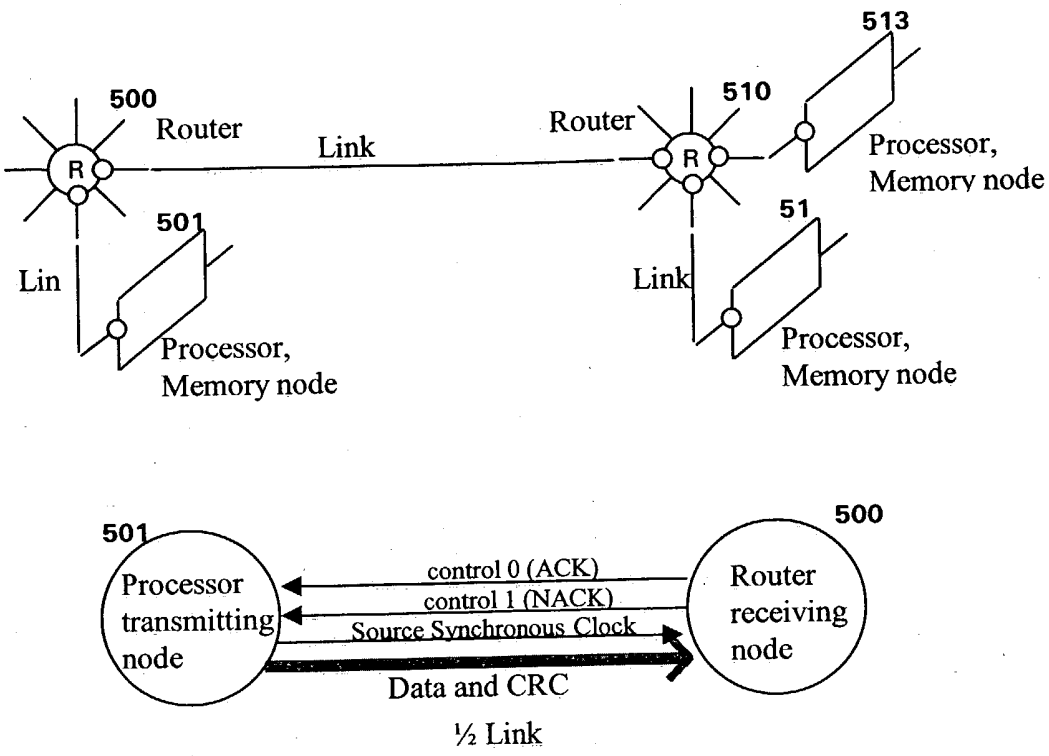


Figure 5

600

byte	8	7	6	5	4	3	2	1	0
	c	d	d	d	d	d	d	d	d
	c	d	d	d	d	d	d	d	d
	c	d	d	d	d	d	d	d	d
	c	d	d	d	d	d	d	d	d
	c	d	d	d	d	d	d	d	d
	c	d	d	d	d	d	d	d	d
	c	d	d	d	d	d	d	d	d
	c	d	d	d	d	d	d	d	d

Data Flow >>>>

Figure 6A

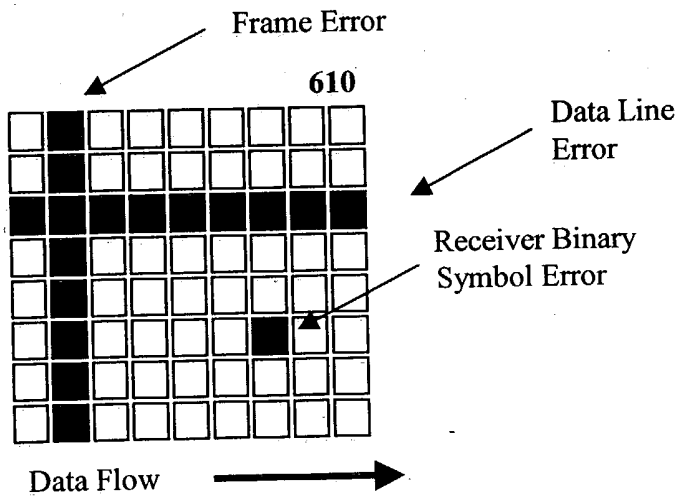


Figure 6B

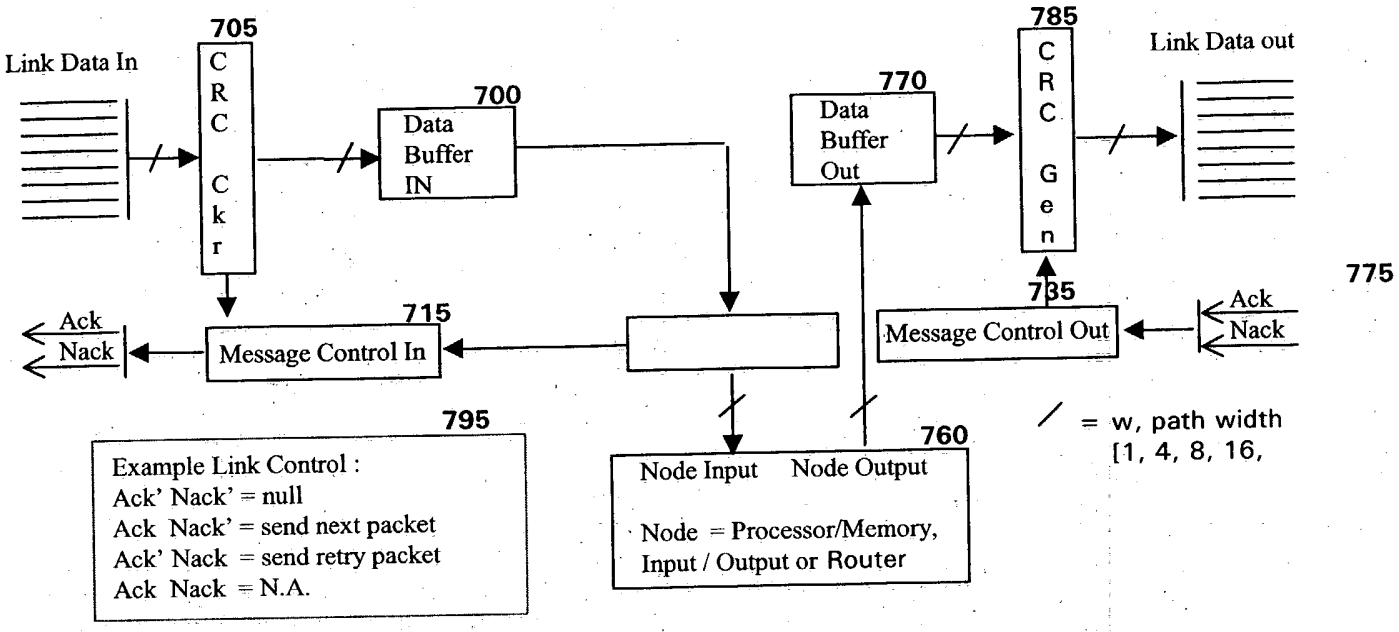

Link Data In
705
CRC Ckr
700
Data Buffer IN
770
Data Buffer Out
785
CRC Gen
Link Data out
775
Ack
Nack
715
Message Control In
735
Message Control Out
Ack
Nack
795
Example Link Control :
Ack' Nack' = null
Ack Nack' = send next packet
Ack' Nack = send retry packet
Ack Nack = N.A.
760
Node Input Node Output
Node = Processor/Memory, Input / Output or Router
/ = w, path width [1, 4, 8, 16,

Figure 7

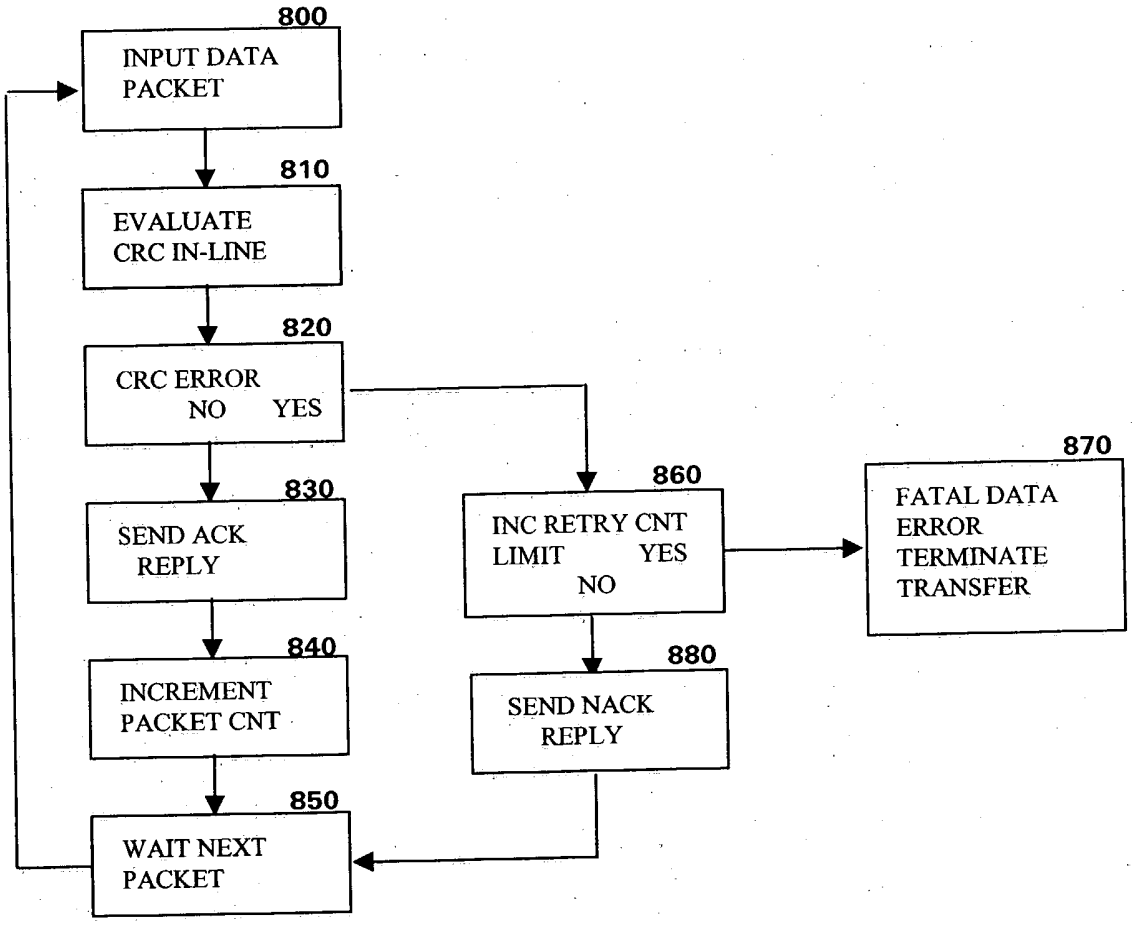

800
INPUT DATA PACKET
810
EVALUATE CRC IN-LINE
820
CRC ERROR
NO YES
830
SEND ACK REPLY
840
INCREMENT PACKET CNT
850
WAIT NEXT PACKET
860
INC RETRY CNT LIMIT
YES
NO
870
FATAL DATA ERROR TERMINATE TRANSFER
880
SEND NACK REPLY

Figure 8

	910										900									
byte	8	7	6	5	4	3	2	1	0		8	7	6	5	4	3	2	1	0	
bit																				
2^0	c	e	e	e	e	e	e	e	1		c	d	d	d	d	d	d	d	1	
2^1	c	e	e	e	e	e	e	e	1		c	d	d	d	d	d	d	d	0	
2^2	c	e	e	e	e	e	e	e	1		c	d	d	d	d	d	d	d	0	
2^3	c	e	e	e	e	e	e	e	0	[E]	c	d	d	d	d	d	d	d	0	[D]
2^4	c	e	e	e	e	e	e	e	1		c	d	d	d	d	d	d	d	0	
2^5	c	e	e	e	e	e	e	e	1		c	d	d	d	d	d	d	d	0	
2^6	c	e	e	e	e	e	e	e	1		c	d	d	d	d	d	d	d	0	
2^7	c	e	e	e	e	e	e	e	0		c	d	d	d	d	d	d	d	1	
						<<< NACK												>>> ACK		

Data Flow >>>>

Figure 9

	1010										1000									
byte	8	7	6	5	4	3	2	1	0		8	7	6	5	4	3	2	1	0	
bit																				
2^0	c	e	e	e	e	e	e	e	1		c	d	d	d	d	d	d	d	1	
2^1	c	e	e	e	e	e	e	e	1		c	d	d	d	d	d	d	d	0	
2^2	c	e	e	e	e	e	e	e	1		c	d	d	d	d	d	d	d	0	
2^3	c	e	e	e	e	e	e	e	0	[E]	c	d	d	d	d	d	d	d	0	[D]
2^4	c	e	e	e	e	e	e	e	1		c	d	d	d	d	d	d	d	0	
2^5	c	e	e	e	e	e	e	e	0		c	d	d	d	d	d	d	d	1	
2^6	c	e	e	e	e	e	e	e	1		c	d	d	d	d	d	d	d	0	
2^7	c	e	e	e	e	e	e	e	0		c	d	d	d	d	d	d	d	1	
						<<< NACK												>>> ACK		

Data Flow >>>>

Figure 10

	ECC	Data
The correction algorithm		
Data byte 0 input in error is		1 0 1 0 0 0 0 1 = A1 hex.
The ECC for A1 is F8	1 1 1 1 1 0 0 0 = F8 hex.	
ECC byte 0 input in error is	0 1 0 1 0 1 1 1 = 57 hex.	
The ECC syndrome	1 0 1 0 1 1 1 1 = AF hex.	
E.P. from Table 5 = d5 & e5	0 0 1 0 0 0 0 0 = e5	and 0 0 1 0 0 0 0 0 = d5
After corrections data = 81 hex.	0 1 1 1 0 1 1 1 = 77 hex.	and 1 0 0 0 0 0 0 1 = 81 hex.

Figure 10 A

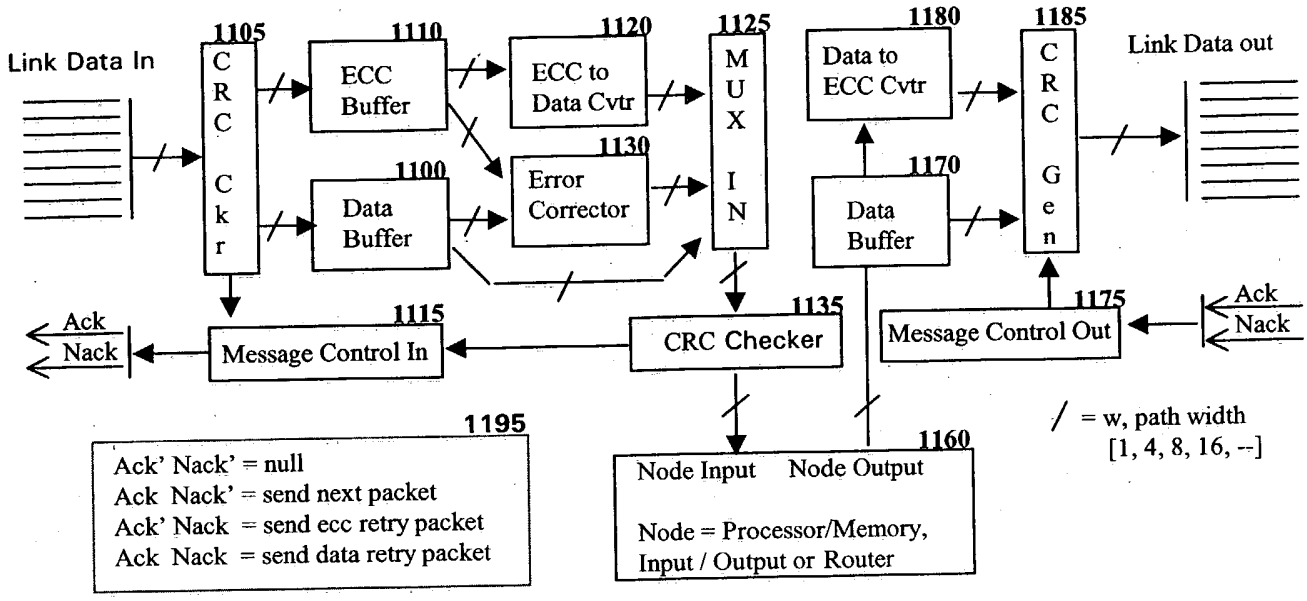

Link Data In
1105
CRC Ckr
1110
ECC Buffer
1120
ECC to Data Cvtr
1125
MUX IN
1180
Data to ECC Cvtr
1185
CRC Gen
Link Data out
1100
Data Buffer
1130
Error Corrector
1170
Data Buffer
Ack
Nack
1115
Message Control In
1135
CRC Checker
1175
Message Control Out
Ack
Nack
1195
Ack' Nack' = null
Ack Nack' = send next packet
Ack' Nack = send ecc retry packet
Ack Nack = send data retry packet
1160
Node Input Node Output
Node = Processor/Memory, Input / Output or Router
/ = w, path width [1, 4, 8, 16, --]

Figure 11

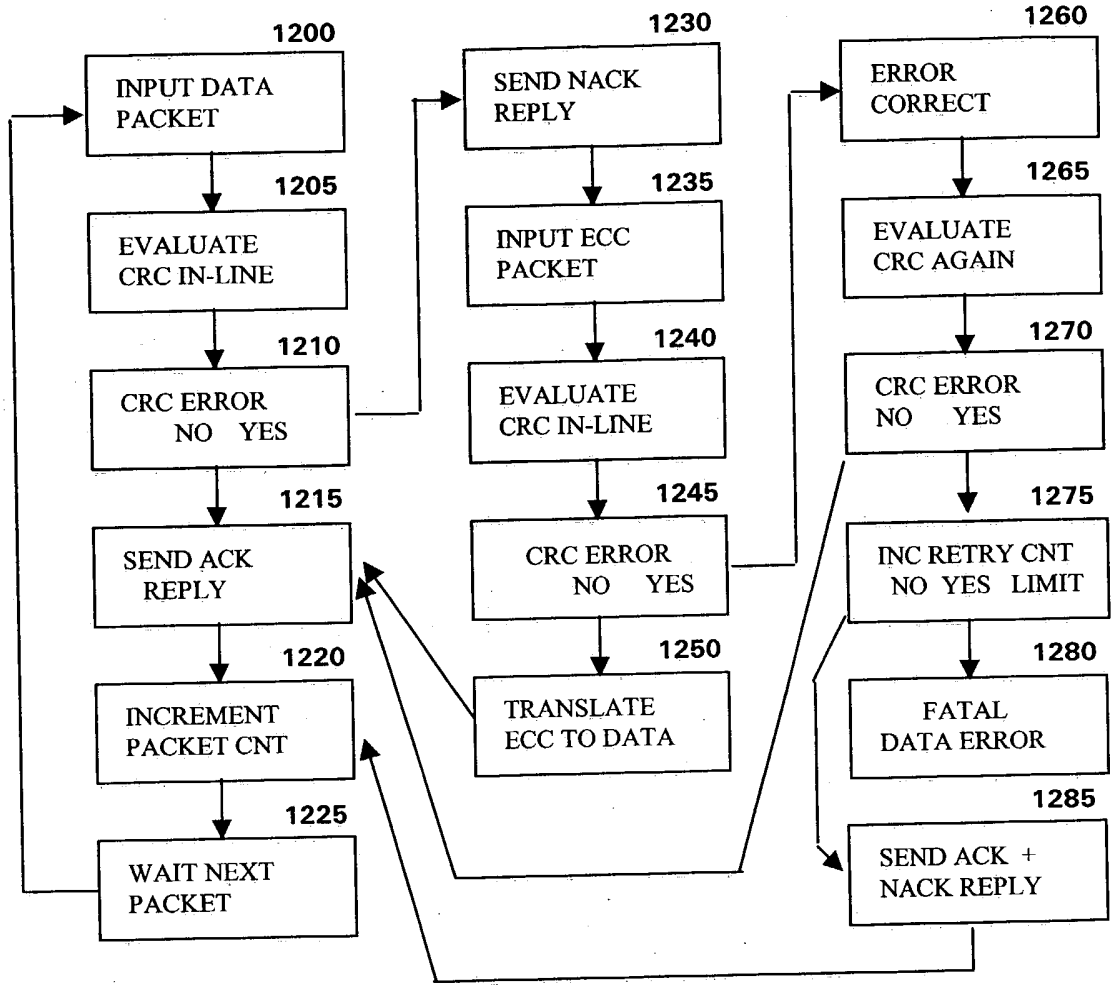

1200
INPUT DATA PACKET
1205
EVALUATE CRC IN-LINE
1210
CRC ERROR NO YES
1215
SEND ACK REPLY
1220
INCREMENT PACKET CNT
1225
WAIT NEXT PACKET
1230
SEND NACK REPLY
1235
INPUT ECC PACKET
1240
EVALUATE CRC IN-LINE
1245
CRC ERROR NO YES
1250
TRANSLATE ECC TO DATA
1260
ERROR CORRECT
1265
EVALUATE CRC AGAIN
1270
CRC ERROR NO YES
1275
INC RETRY CNT NO YES LIMIT
1280
FATAL DATA ERROR
1285
SEND ACK + NACK REPLY

Figure 12

[E]								1301	[D]								1300
c6uc2u	u30	u26	u22	u18	u14	u10	u06	u02	c4Sc0S	s28	s24	s20	s16	s12	s08	s04	s00
c6vc2v	v30	v26	v22	v18	v14	v10	v06	v02	c4tc0t	t28	t24	t20	t16	t12	t08	t04	t00
c7uc3u	u31	u27	u23	u19	u15	u11	u07	u03	c5sc1s	s29	s25	s21	s17	s13	s09	s05	s01
c7vc3v	v31	v27	v23	v19	v15	v11	v07	v03	c5tc1t	t29	t25	t21	t17	t13	t09	t05	t01
c4uc0u	u28	u24	u20	u16	u12	u08	u04	u00	c6sc2s	s30	s26	s22	s18	s14	s10	s06	s02
c4vc0v	v28	v24	v20	v16	v12	v08	v04	v00	c6tc2t	t30	t26	t22	t18	t14	t10	t06	t02
c5uc1u	u29	u25	u21	u17	u13	u09	u05	u01	c7sc3s	s31	s27	s23	s19	s15	s11	s07	s03
c5vc1v	v29	v25	v21	v17	v13	v09	v05	v01	c7tc3t	t31	t27	t23	t19	t15	t11	t07	t03

Data Flow >>> <<< NACK <<< ACK

Figure 13

[E]								1401	[D]								1400
c6uc2u	u30	u26	u22	u18	u14	u10	u06	u02	c4Sc0S	s28	s24	s20	s16	s12	s08	s04	~~s00~~
c6vc2v	v30	v26	v22	v18	v14	v10	v06	v02	c4tc0t	t28	t24	t20	t16	t12	t08	t04	t00
c7uc3u	u31	u27	u23	u19	u15	u11	u07	u03	c5sc1s	s29	s25	s21	s17	s13	s09	s05	s01
c7vc3v	v31	v27	v23	v19	v15	v11	v07	v03	c5tc1t	t29	t25	t21	t17	t13	t09	t05	t01
c4uc0u	u28	u24	u20	u16	u12	u08	u04	~~u00~~	c6sc2s	s30	s26	s22	s18	s14	s10	s06	s02
c4vc0v	v28	v24	v20	v16	v12	v08	v04	v00	c6tc2t	t30	t26	t22	t18	t14	t10	t06	t02
c5uc1u	u29	u25	u21	u17	u13	u09	u05	u01	c7sc3s	s31	s27	s23	s19	s15	s11	s07	s03
c5vc1v	v29	v25	v21	v17	v13	v09	v05	v01	c7tc3t	t31	t27	t23	t19	t15	t11	t07	t03

Data Flow >>> <<< NACK <<< ACK

Figure 14

Begin with Byte 00
Transmitted
s00t00 = 18h (data = ts = 81h) u00v00 = 77h (ECC = vu = 77h)
Received
s00t00 = 1Ah (data = ts = A1h) u00v00 = 75h (ECC = vu = 57h)
So correction proceeds exactly as before in Figure 8 for byte 00.
All 32 bytes are assembled and corrected then verified via the CRC checkcode comparison.

	ECC	Data
Data byte 00 input in error is		1 0 1 0 0 0 0 1 = A1 hex.
The ECC for A1 is F8	1 1 1 1 1 0 0 0 = F8 hex.	
ECC byte 00 input in error is	0 1 0 1 0 1 1 1 = 57 hex.	
The ECC syndrome	1 0 1 0 1 1 1 1 = AF hex.	
E.P. from Table 1 = d5 & e5	0 0 1 0 0 0 0 0 = e5	and 0 0 1 0 0 0 0 0 = d5
After corrections data = 81 hex.	0 1 1 1 0 1 1 1 = 77 hex.	and 1 0 0 0 0 0 0 1 = 81 hex.

Figure 14A

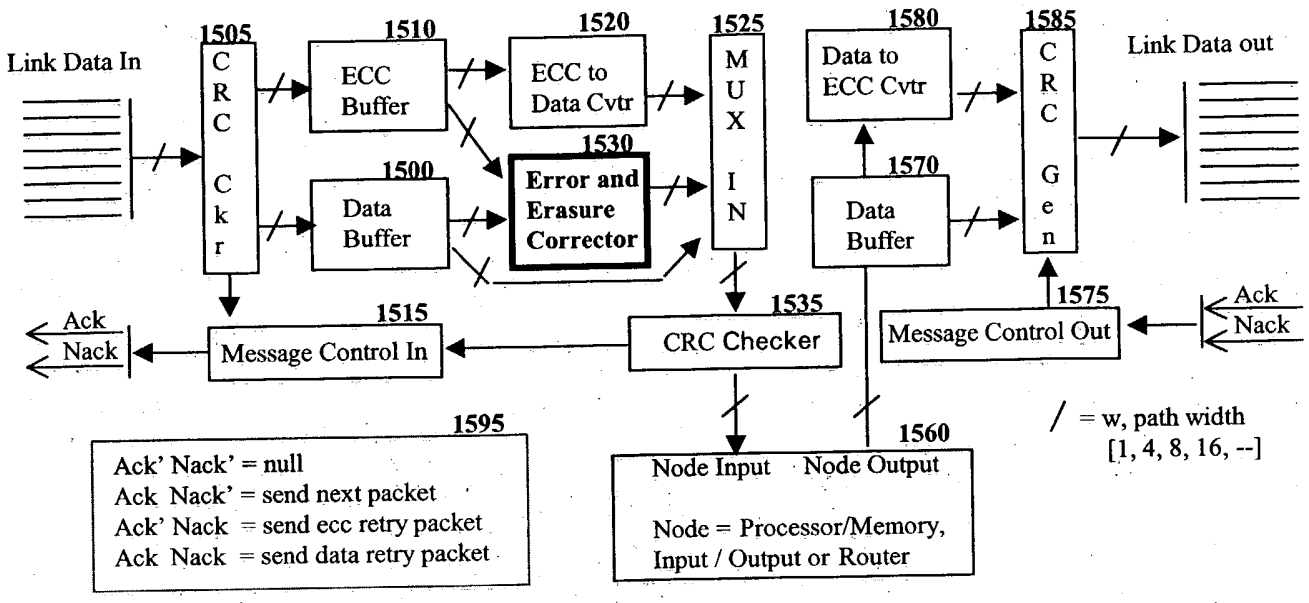

Link Data In
1505
CRC Ckr
1510
ECC Buffer
1520
ECC to Data Cvtr
1525
MUX IN
1530
Error and Erasure Corrector
1500
Data Buffer
1580
Data to ECC Cvtr
1585
CRC Gen
Link Data out
1570
Data Buffer
Ack
Nack
1515
Message Control In
1535
CRC Checker
1575
Message Control Out
Ack
Nack
/ = w, path width
[1, 4, 8, 16, --]
1595
Ack' Nack' = null
Ack Nack' = send next packet
Ack' Nack = send ecc retry packet
Ack Nack = send data retry packet
1560
Node Input Node Output
Node = Processor/Memory,
Input / Output or Router

Figure 15

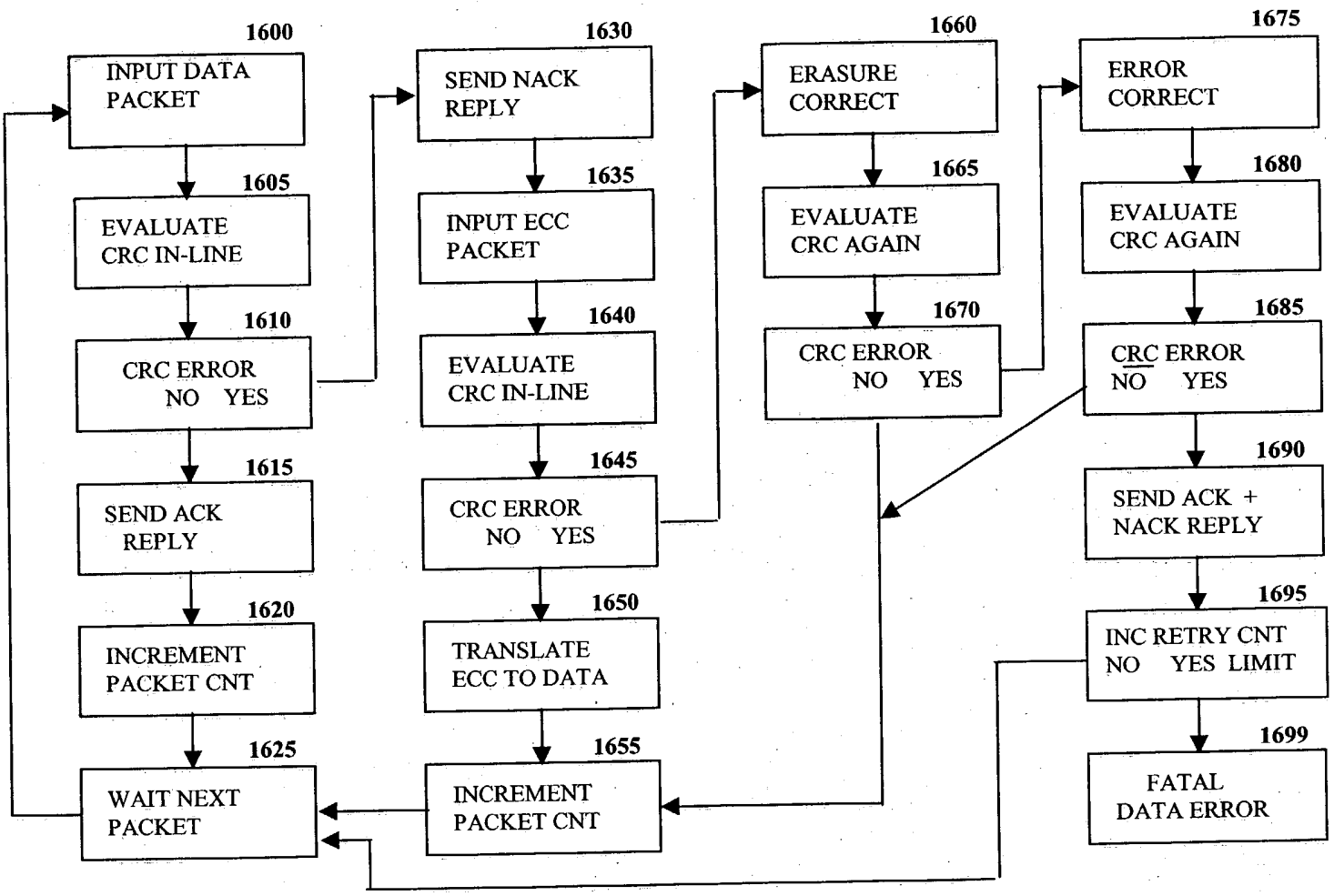
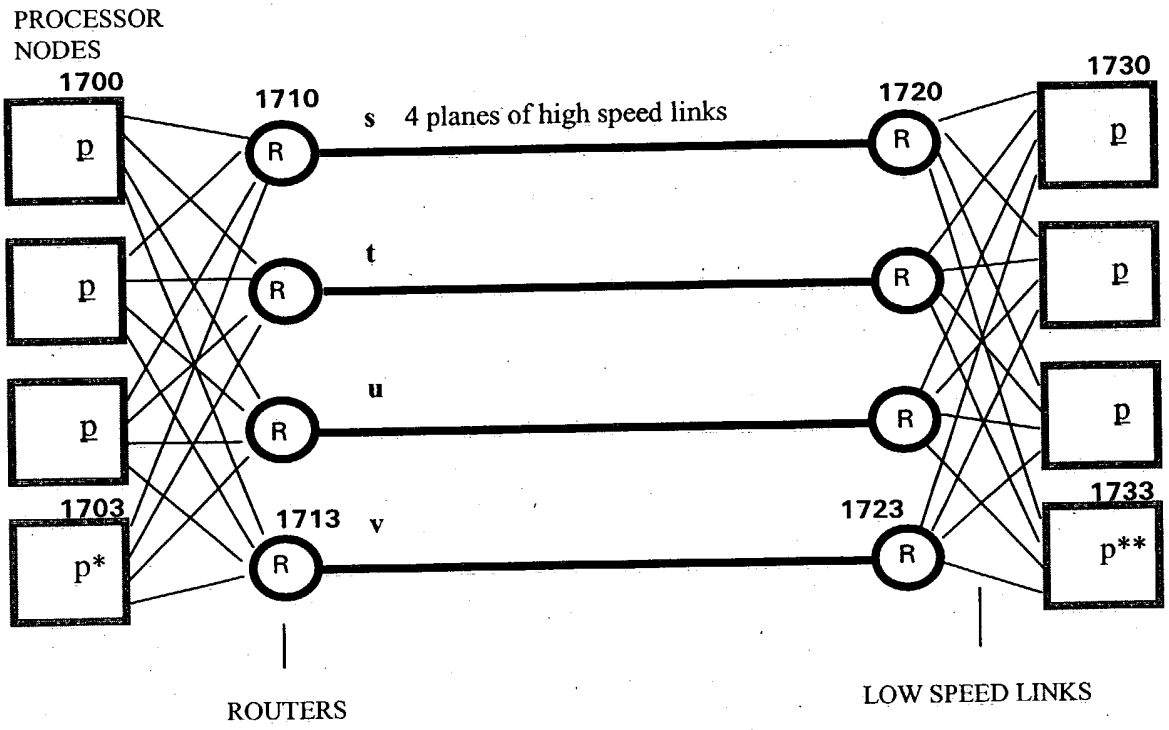

1600
INPUT DATA PACKET
1605
EVALUATE CRC IN-LINE
1610
CRC ERROR NO YES
1615
SEND ACK REPLY
1620
INCREMENT PACKET CNT
1625
WAIT NEXT PACKET
1630
SEND NACK REPLY
1635
INPUT ECC PACKET
1640
EVALUATE CRC IN-LINE
1645
CRC ERROR NO YES
1650
TRANSLATE ECC TO DATA
1655
INCREMENT PACKET CNT
1660
ERASURE CORRECT
1665
EVALUATE CRC AGAIN
1670
CRC ERROR NO YES
1675
ERROR CORRECT
1680
EVALUATE CRC AGAIN
1685
CRC ERROR NO YES
1690
SEND ACK + NACK REPLY
1695
INC RETRY CNT NO YES LIMIT
1699
FATAL DATA ERROR

Figure 16



$[D_i]$ $d^0d^1d^2d^3d^4d^5d^6d^7$
= st

$[E_i]$ $e^0e^1e^2e^3e^4e^5e^6e^7$
= uv

$d^0d^1d^2d^3$ **s**

$d^4d^5d^6d^7$ **t**

$e^0e^1e^2e^3$ **u**

$e^4e^5e^6e^7$ **v**

su
sv **st** **st** = $[D_i]$
tu **uv**
tv
uv

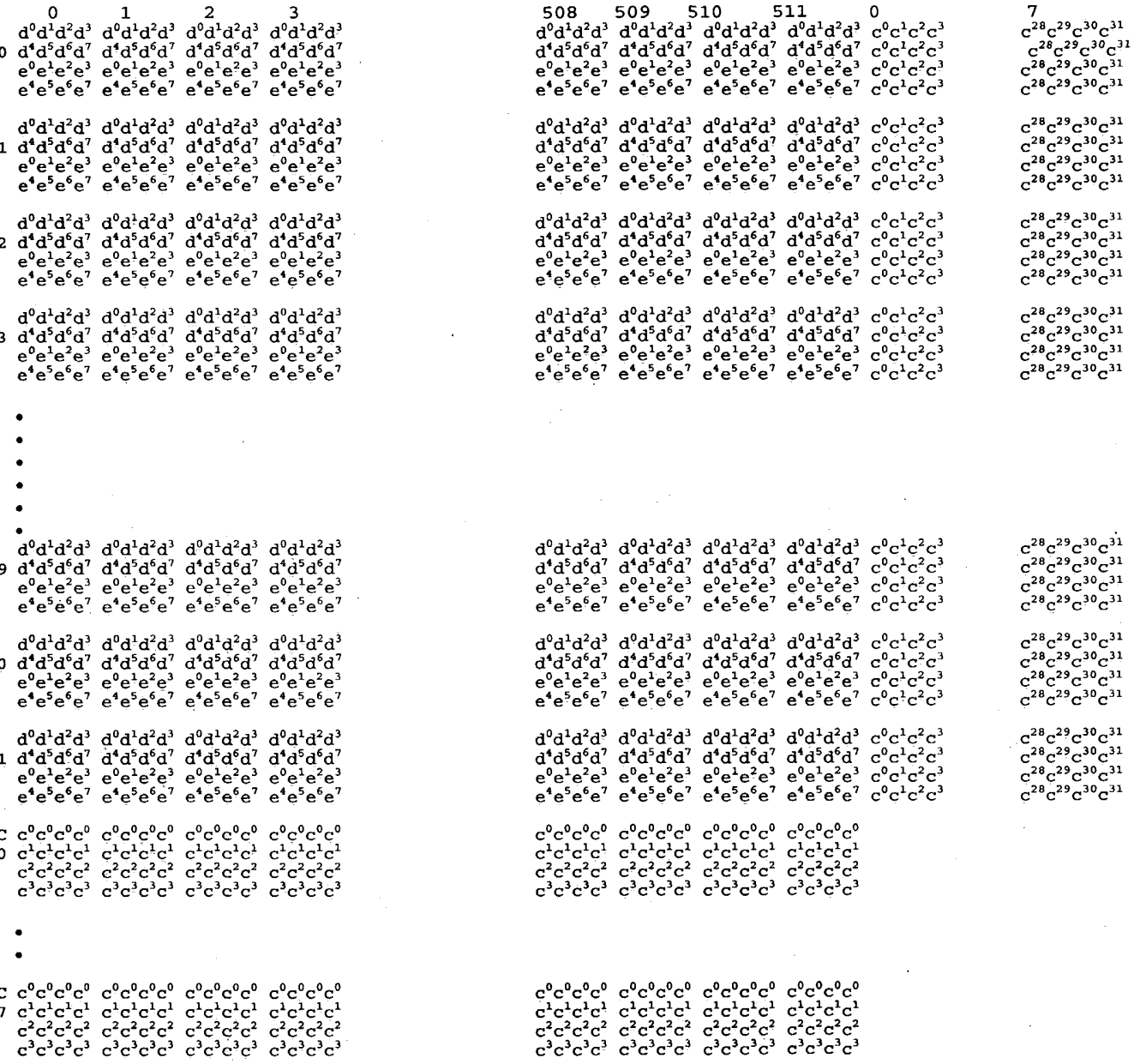
data encoded ECC

data dispersed across 4 planes

data assembly and recovery

data verification with and correction

Figure 17



Array Size = 512 codewords by 512 codewords = 262,144 Bytes
ECC Size = 262,144 Bytes
CRC Size = 512 by 2 edges by 8 CWs by 2 bytes/CW = 16,384 Bytes
Efficiency = 262,144 / [(262, 144 x 2) + 16,384] = 0.4848

Figure 18

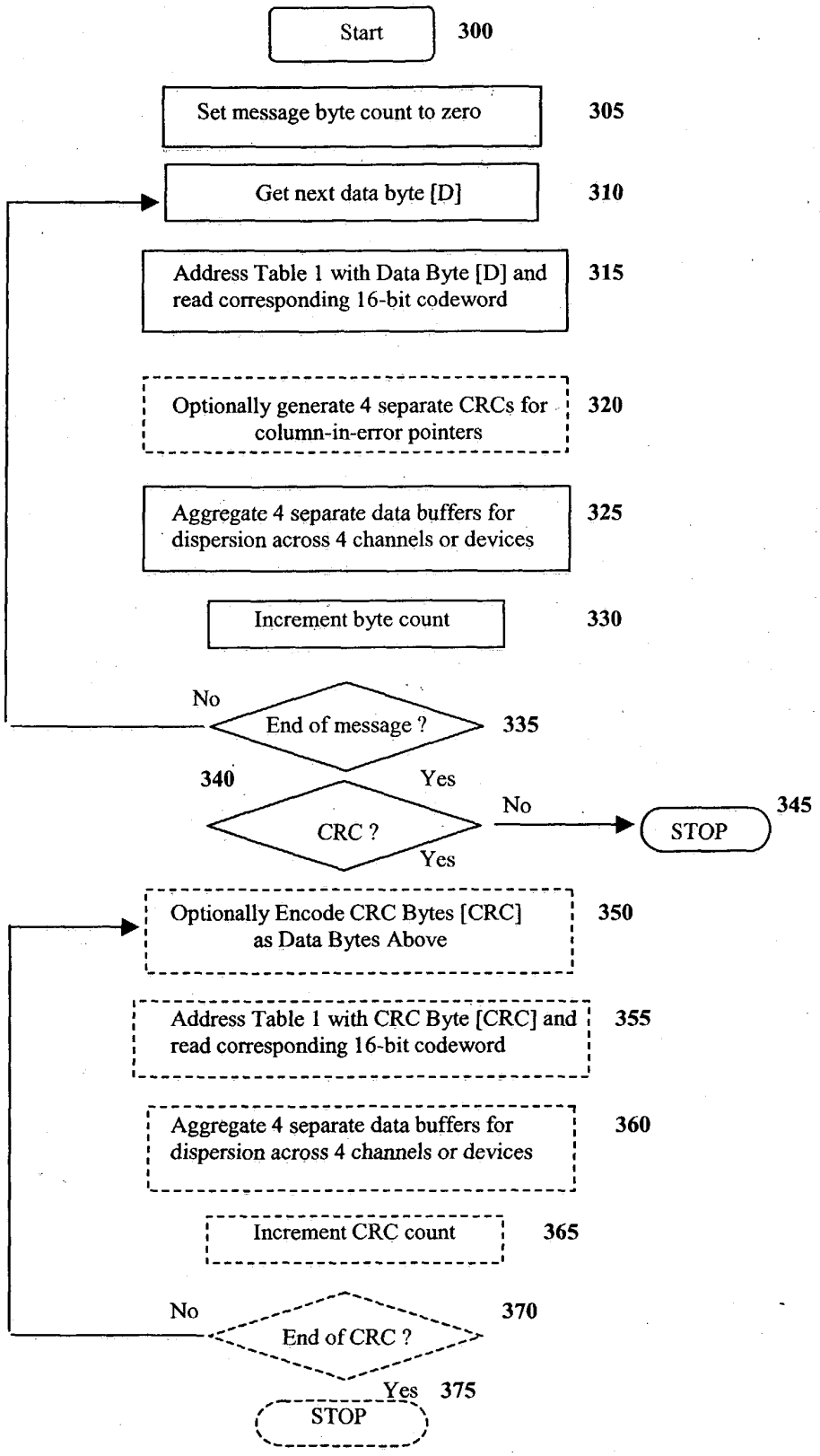


Figure 19 Flowchart for encoding codeword arrays

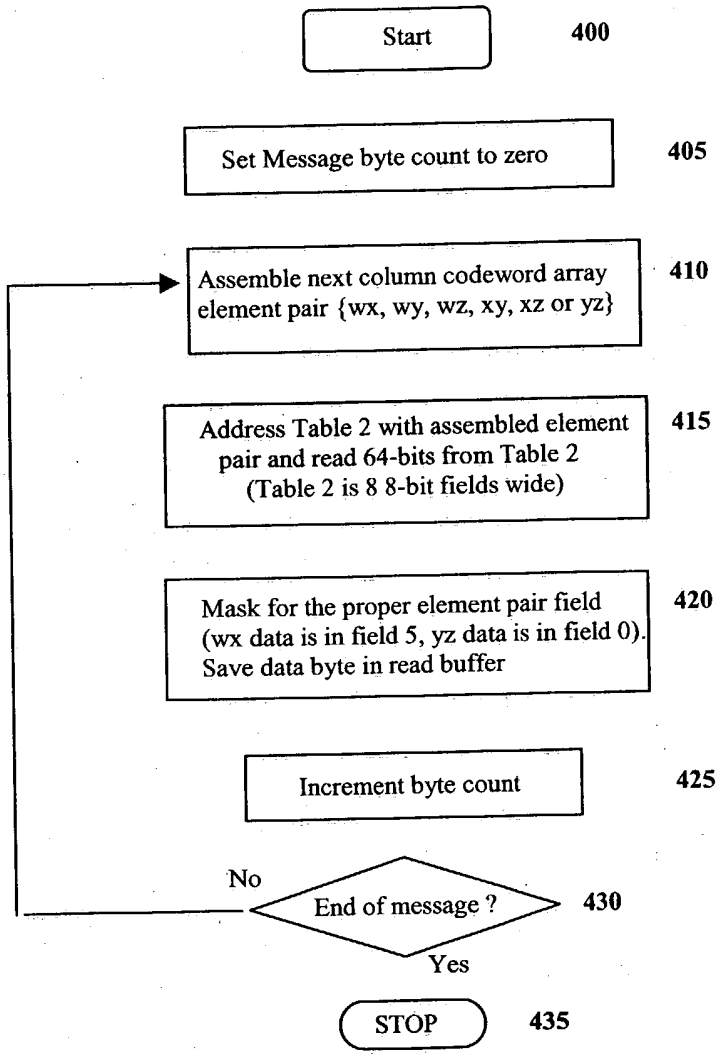


Figure 20 Flowchart for decoding codewords

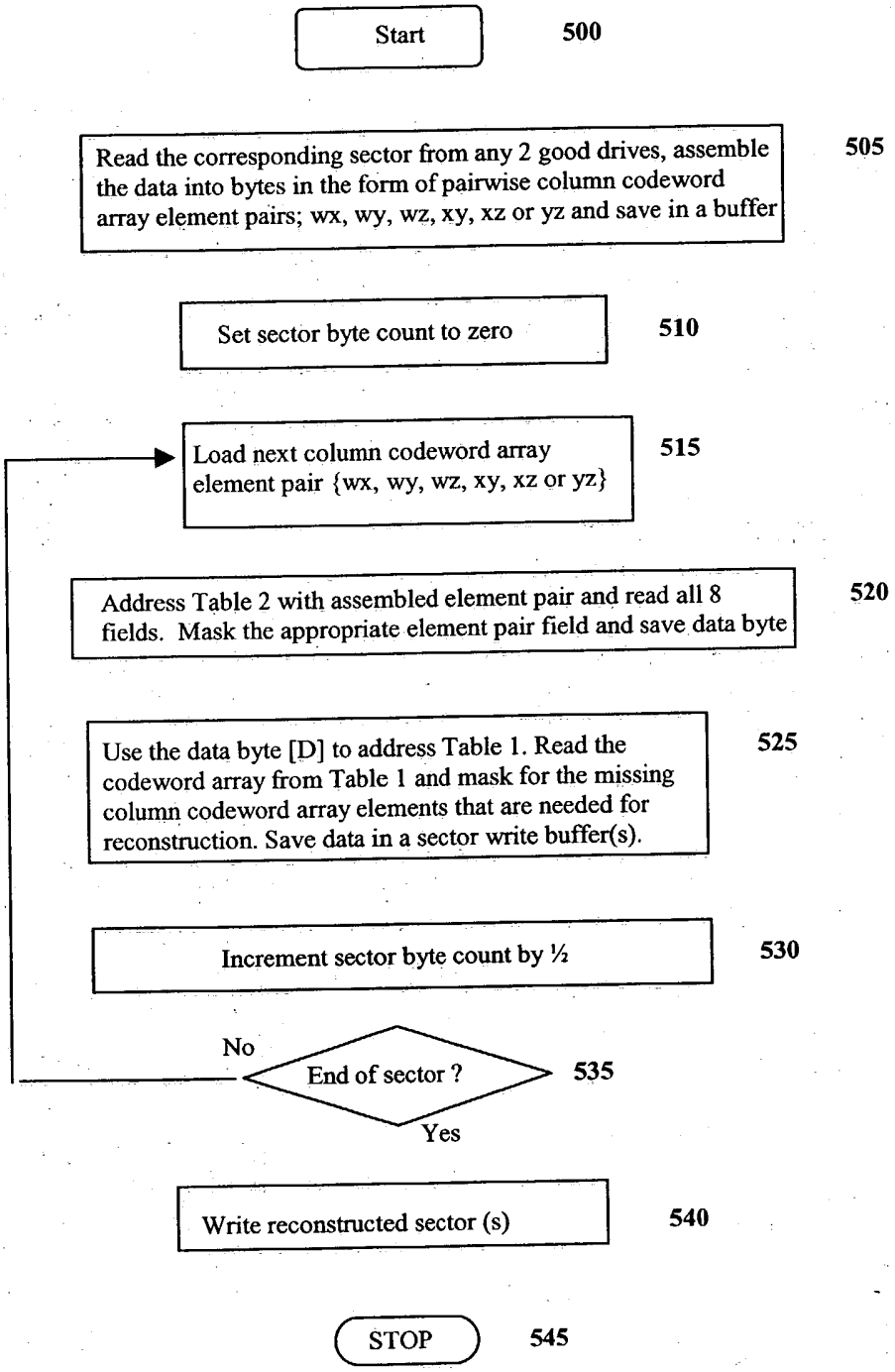


Figure 21 Flowchart for reconstructing a disk drive sector

Table 1 Codewords denoted as column elements
The table codewords is based on column values using the ECC polynomial
g1(x) = 1 + x3 + x4 + x5 + x8 . A similar table shows the codewords based on row values.

The codeword [CWi]= [Di] [Ei] = d0d1d2d3d4d5d6d7e0e1e2e3e4e5e6e7
The codeword in binary array form:

```
[Di]   d0d1d2d3 s {hexadecimal row dispersal}
       d4d5d6d7 t
[Ei]   e0e1e2e3 u
       e4e5e6e7 v
      ---------
        w x y z    {hexadecimal column dispersal}
```

Codeword packet generation {all values are hexadecimal}

Data	cp	Data	cp	Data	cp	Data	cp	Data	cp	Data	cp	Data	cp	Data	cp
00	0000	01	D804	02	8D80	03	5584	04	08D8	05	D0DC	06	8558	07	5D5C
08	C889	09	108D	0A	4509	0B	9D0D	0C	C051	0D	1855	0E	4DD1	0F	95D5
10	E48C	11	3C88	12	690C	13	B108	14	EC54	15	3450	16	61D4	17	B9D0
18	2C05	19	F401	1A	A185	1B	7981	1C	24DD	1D	FCD9	1E	A95D	1F	7159
20	464C	21	9E48	22	CBCC	23	13C8	24	4E94	25	9690	26	C314	27	1B10
28	8EC5	29	56C1	2A	0345	2B	DB41	2C	861D	2D	5E19	2E	0B9D	2F	D399
30	A2C0	31	7AC4	32	2F40	33	F744	34	AA18	35	721C	36	2798	37	FF9C
38	6A49	39	B24D	3A	E7C9	3B	3FCD	3C	6291	3D	BA95	3E	EF11	3F	3715
40	4C60	41	9464	42	C1E0	43	19E4	44	44B8	45	9CBC	46	C938	47	113C
48	84E9	49	5CED	4A	0969	4B	D16D	4C	8C31	4D	5435	4E	01B1	4F	D9B5
50	A8EC	51	70E8	52	256C	53	FD68	54	A034	55	7830	56	2DB4	57	F5B0
58	6065	59	B861	5A	EDE5	5B	35E1	5C	68BD	5D	B0B9	5E	E53D	5F	3D39
60	0A2C	61	D228	62	87AC	63	5FA8	64	02F4	65	DAF0	66	8F74	67	5770
68	C2A5	69	1AA1	6A	4F25	6B	9721	6C	CA7D	6D	1279	6E	47FD	6F	9FF9
70	EEA0	71	36A4	72	6320	73	BB24	74	E678	75	3E7C	76	6BF8	77	B3FC
78	2629	79	FE2D	7A	ABA9	7B	73AD	7C	2EF1	7D	F6F5	7E	A371	7F	7B75
80	04C6	81	DCC2	82	8946	83	5142	84	0C1E	85	D41A	86	819E	87	599A
88	CC4F	89	144B	8A	41CF	8B	99CB	8C	C497	8D	1C93	8E	4917	8F	9113
90	E04A	91	384E	92	6DCA	93	B5CE	94	E892	95	3096	96	6512	97	BD16
98	28C3	99	F0C7	9A	A543	9B	7D47	9C	201B	9D	F81F	9E	AD9B	9F	759F
A0	428A	A1	9A8E	A2	CF0A	A3	170E	A4	4A52	A5	9256	A6	C7D2	A7	1FD6
A8	8A03	A9	5207	AA	0783	AB	DF87	AC	82DB	AD	5ADF	AE	0F5B	AF	D75F
B0	A606	B1	7E02	B2	2B86	B3	F382	B4	AEDE	B5	76DA	B6	235E	B7	FB5A
B8	6E8F	B9	B68B	BA	E30F	BB	3B0B	BC	6657	BD	BE53	BE	EBD7	BF	33D3
C0	48A6	C1	90A2	C2	C526	C3	1D22	C4	407E	C5	987A	C6	CDFE	C7	15FA
C8	802F	C9	582B	CA	0DAF	CB	D5AB	CC	88F7	CD	50F3	CE	0577	CF	DD73
D0	AC2A	D1	742E	D2	21AA	D3	F9AE	D4	A4F2	D5	7CF6	D6	2972	D7	F176
D8	64A3	D9	BCA7	DA	E923	DB	3127	DC	6C7B	DD	B47F	DE	E1FB	DF	39FF
E0	0EEA	E1	D6EE	E2	836A	E3	5B6E	E4	0632	E5	DE36	E6	8BB2	E7	53B6
E8	C663	E9	1E67	EA	4BE3	EB	93E7	EC	CEBB	ED	16BF	EE	433B	EF	9B3F
F0	EA66	F1	3262	F2	67E6	F3	BFE2	F4	E2BE	F5	3ABA	F6	6F3E	F7	B73A
F8	22EF	F9	FAEB	FA	AF6F	FB	776B	FC	2A37	FD	F233	FE	A7B7	FF	7FB3

Table 2 - Column Decode Table (abbreviated)

Decoding table for columns. {all values are hexadecimal}
Data In outputs ECC in Frame 7
ECC in outputs Data in Frame 6
Couplet wx, wy, wz, xy, xz and yz in outputs Data in Frames 0 - 5

Byte number	7	6	5	4	3	2	1	0
Addr. = {dat	ECC	DATA	DATA	DATA	DATA	DATA	DATA	DATA
ecc, cpq pair	{addr = data	{addr = ecc}	{addr = wx}	{addr = wy}	{addr = wz}	{addr = xy}	{addr = xz}	{addr = yz}
00	00	00	00	00	00	00	00	00
01	39	72	4E	84	4E	C8	0C	19
02	72	E4	64	60	E4	9C	C1	B1
03	4B	96	2A	E4	AA	54	CD	A8
04	E4	F1	80	2A	64	90	54	01
05	DD	83	CE	AE	2A	0C	58	18
06	96	15	E4	4A	80	58	95	B0
07	AF	67	AA	CE	CE	C4	99	A9
08	F1	DB	04	AA	04	09	51	13
09	C8	A9	4A	2E	4A	95	5D	0A
0A	83	3F	60	CA	E0	C1	90	A2
0B	BA	4D	2E	4E	AE	5D	96	BB
0C	15	2A	84	80	60	99	05	12
0D	2C	58	CA	04	2E	05	09	0B
0E	67	CE	E0	E0	84	57	C4	A3
0F	5E	BC	AE	64	CA	CD	C8	C8
10	DB	8F	09	A3	27	13	42	27
11	E2	FD	47	27	69	8F	4E	3E
12	A9	6B	6D	C3	C3	DB	83	96
1F	85	33	A7	C7	ED	DE	8A	9D
43	6C	D8	EE	EE	EA	40	D8	9A
47	88	29	6E	C4	8E	D0	8C	9B
9C	80	39	45	8B	45	8B	12	37
D2	C0	B9	61	61	81	C3	C3	A6
E5	3B	76	5E	14	5A	BD	28	5A
E6	70	E0	74	F0	F0	E9	E5	F2
E7	49	92	3A	74	BE	75	E9	EB
E8	17	2E	94	10	74	B8	21	51
E9	2E	5C	DA	94	3A	24	2D	48
EA	65	CA	F0	70	90	70	E0	E0
EB	5C	B8	BE	F4	DE	EC	EC	F9
EC	F3	DF	14	3A	10	28	75	50
ED	CA	AD	5A	BE	5E	B4	79	49
EE	81	3B	70	5A	F4	E0	B4	E1
EF	B8	49	3E	DE	BA	7C	B8	F8
F0	3D	7A	99	19	57	A2	32	65
F1	04	08	D7	9D	19	F1	F1	7C
F2	4F	9E	FD	79	B3	6A	F3	D4
F3	76	EC	B3	FD	FD	F6	FF	CD
F4	D9	8B	19	33	33	32	66	64
F5	E0	F9	57	B7	7D	AE	6A	7D
F6	AB	6F	7D	53	D7	FA	A7	D5
F7	92	1D	33	D7	99	66	AB	CC
F8	CC	A1	9D	B3	53	AB	63	76
F9	F5	D3	D3	37	1D	37	6F	6F
FA	BE	45	F9	D3	B7	63	A2	C7
FB	87	37	B7	57	F9	FF	AE	DE
FC	28	50	1D	99	37	3B	37	77
FD	11	22	53	1D	79	A7	3B	6E
FE	5A	B4	79	F9	D3	F3	F6	C6
FF	63	C6	37	7D	9D	6F	FA	DF

Table 3 Codewords denoted as row elements
The table codewords is based on row values using the ECC polynomial
$g_l(x) = 1 + x^3 + x^4 + x^5 + x^8$. A similar table shows the codewords based on column values.

The codeword $[CW_i]= [D_i]\,[E_i] = d^0d^1d^2d^3d^4d^5d^6d^7e^0e^1e^2e^3e^4e^5e^6e^7$
The codeword in binary array form:

$[D_i]$ $d^0d^1d^2d^3$ **s** {hexadecimal row dispersal}
$d^4d^5d^6d^7$ **t**
$[E_i]$ $e^0e^1e^2e^3$ **u**
$e^4e^5e^6e^7$ **v**
---
w x y z {hexadecimal column dispersal}

Codeword packet generation {all values are hexadecimal}

Data	cp	Data	cp	Data	cp	Data	cp	Data	cp	Data	cp	Data	cp	Data	cp
00	0000	01	1093	02	2027	03	30B4	04	404E	05	50DD	06	6069	07	70FA
08	801F	09	908C	0A	A038	0B	B0AB	0C	C051	0D	D0C2	0E	E076	0F	F0E5
10	01BD	11	112E	12	219A	13	3109	14	41F3	15	5160	16	61D4	17	7147
18	81A2	19	9131	1A	A185	1B	B116	1C	C1EC	1D	D17F	1E	E1CB	1F	F158
20	02F8	21	126B	22	22DF	23	324C	24	42B6	25	5225	26	6291	27	7202
28	82E7	29	9274	2A	A2C0	2B	B253	2C	C2A9	2D	D23A	2E	E28E	2F	F21D
30	0345	31	13D6	32	2362	33	33F1	34	430B	35	5398	36	632C	37	73BF
38	835A	39	93C9	3A	A37D	3B	B3EE	3C	C314	3D	D387	3E	E333	3F	F3A0
40	0472	41	14E1	42	2455	43	34C6	44	443C	45	54AF	46	641B	47	7488
48	846D	49	94FE	4A	A44A	4B	B4D9	4C	C423	4D	D4B0	4E	E404	4F	F497
50	05CF	51	155C	52	25E8	53	357B	54	4581	55	5512	56	65A6	57	7535
58	85D0	59	9543	5A	A5F7	5B	B564	5C	C59E	5D	D50D	5E	E5B9	5F	F52A
60	068A	61	1619	62	26AD	63	363E	64	46C4	65	5657	66	66E3	67	7670
68	8695	69	9606	6A	A6B2	6B	B621	6C	C6DB	6D	D648	6E	E6FC	6F	F66F
70	0737	71	17A4	72	2710	73	3783	74	4779	75	57EA	76	675E	77	77CD
78	8728	79	97BB	7A	A70F	7B	B79C	7C	C766	7D	D7F5	7E	E741	7F	F7D2
80	08E4	81	1877	82	28C3	83	3850	84	48AA	85	5839	86	688D	87	781E
88	88FB	89	9868	8A	A8DC	8B	B84F	8C	C8B5	8D	D826	8E	E892	8F	F801
90	0959	91	19CA	92	297E	93	39ED	94	4917	95	5984	96	6930	97	79A3
98	8946	99	99D5	9A	A961	9B	B9F2	9C	C908	9D	D99B	9E	E92F	9F	F9BC
A0	0A1C	A1	1A8F	A2	2A3B	A3	3AA8	A4	4A52	A5	5AC1	A6	6A75	A7	7AE6
A8	8A03	A9	9A90	AA	AA24	AB	BAB7	AC	CA4D	AD	DADE	AE	EA6A	AF	FAF9
B0	0BA1	B1	1B32	B2	2B86	B3	3B15	B4	4BEF	B5	5B7C	B6	6BC8	B7	7B5B
B8	8BBE	B9	9B2D	BA	AB99	BB	BB0A	BC	CBF0	BD	DB63	BE	EBD7	BF	FB44
C0	0C96	C1	1C05	C2	2CB1	C3	3C22	C4	4CD8	C5	5C4B	C6	6CFF	C7	7C6C
C8	8C89	C9	9C1A	CA	ACAE	CB	BC3D	CC	CCC7	CD	DC54	CE	ECE0	CF	FC73
D0	0D2B	D1	1DB8	D2	2D0C	D3	3D9F	D4	4D65	D5	5DF6	D6	6D42	D7	7DD1
D8	8D34	D9	9DA7	DA	AD13	DB	BD80	DC	CD7A	DD	DDE9	DE	ED5D	DF	FDCE
E0	0E6E	E1	1EFD	E2	2E49	E3	3EDA	E4	4E20	E5	5EB3	E6	6E07	E7	7E94
E8	8E71	E9	9EE2	EA	AE56	EB	BEC5	EC	CE3F	ED	DEAC	EE	EE18	EF	FE8B
F0	0FD3	F1	1F40	F2	2FF4	F3	3F67	F4	4F9D	F5	5F0E	F6	6FBA	F7	7F29
F8	8FCC	F9	9F5F	FA	AFEB	FB	BF78	FC	CF82	FD	DF11	FE	EFA5	FF	FF36

Table 4 - Row Decode Table (abbreviated)

Decoding table for rows. {all values are hexadecimal}

Data In outputs ECC in Frame 7

ECC in outputs Data in Frame 6

Couplet st, su, sv, tu, tv or uv in outputs Data in Frames 0 - 5

Byte number	7	6	5	4	3	2	1	0
Addr. = {data, ecc, cpq pair}	ECC {addr = data}	DATA {addr = ecc}	DATA {addr = st}	DATA {addr = su}	DATA {addr = sv}	DATA {addr = tu}	DATA {addr = tv}	DATA {addr = uv}
00	00	00	00	00	00	00	00	00
01	39	72	10	A0	B0	08	0C	8F
02	72	E4	20	D0	40	02	0D	27
03	4B	96	30	70	F0	0A	01	A8
04	E4	F1	40	30	80	04	03	4E
05	DD	83	50	90	30	0C	0F	C1
06	96	15	60	E0	C0	06	0E	69
07	AF	67	70	40	70	0E	02	E6
08	F1	DB	80	60	20	09	0A	9C
09	C8	A9	90	C0	90	01	06	13
0A	83	3F	A0	B0	60	0B	07	BB
0B	BA	4D	BO	10	D0	03	0B	34
0C	15	2A	C0	50	A0	0D	09	D2
0D	2C	58	D0	F0	10	05	05	5D
0E	67	CE	E0	80	E0	0F	04	F5
0F	5E	BC	F0	20	50	07	08	7A
10	DB	8F	01	C1	F1	13	15	72
11	E2	FD	11	61	41	1B	19	FD
12	A9	6B	21	11	B1	11	18	55
1F	85	33	F1	E1	A1	14	1D	08
43	6C	D8	34	44	14	44	4C	59
47	88	29	74	74	94	40	47	E2
9C	80	39	C9	39	09	91	9F	7B
D2	C0	B9	2D	8D	0D	D0	D6	7F
E5	3B	76	5E	DE	FE	EA	EB	0F
E6	70	E0	6E	AE	0E	E0	EA	A7
E7	49	92	7E	0E	BE	E8	E6	28
E8	17	2E	8E	2E	EE	EF	EE	52
E9	2E	5C	9E	8E	5E	E7	E2	DD
EA	65	CA	AE	FE	AE	ED	E3	75
EB	5C	B8	BE	5E	1E	E5	EF	FA
EC	F3	DF	CE	1E	6E	EB	ED	1C
ED	CA	AD	DE	BE	DE	E3	E1	93
EE	81	3B	EE	CE	2E	E9	E0	3B
EF	B8	49	FE	6E	9E	E1	EC	B4
F0	3D	7A	0F	8F	3F	F5	F1	BC
F1	04	08	1F	2F	8F	FD	FD	33
F2	4F	9E	2F	5F	7F	F7	FC	9B
F3	76	EC	3F	FF	CF	FF	F0	14
F4	D9	8B	4F	BF	BF	F1	F2	F2
F5	E0	F9	5F	1F	0F	F9	FE	7D
F6	AB	6F	6F	6F	FF	F3	FF	D5
F7	92	1D	7F	CF	4F	FB	F3	5A
F8	CC	A1	8F	EF	1F	FC	FB	20
F9	F5	D3	9F	4F	AF	F4	F7	AF
FA	BE	45	AF	3F	5F	FE	F6	07
FB	87	37	BF	9F	EF	F6	FA	88
FC	28	50	CF	DF	9F	F8	F8	6E
FD	11	22	DF	7F	2F	F0	F4	E1
FE	5A	B4	EF	0F	DF	FA	F5	49
FF	63	C6	FF	AF	6F	F2	F9	C6

Table 5 Error Pattern Table

1 or 2 bit error patterns for polynomial $g_1(x) = 1 + x^3 + x^4 + x^5 + x^8$ addressed by error syndrome values {syndrome values are hexadecimal}. There are 16 single bit errors and 120 double bit errors. For actual use the table values would have error pattern bits set to ones that would toggle the data and ecc bits via the ex-or logical operation. It is also possible to simply translate the results via hardware logical operations.

The codeword $[CW_i]= [D_i]\ [E_i] = d^0d^1d^2d^3d^4d^5d^6d^7e^0e^1e^2e^3e^4e^5e^6e^7$

D0D1D2D3D4D5D6D7E0E1E2E3E4E5E6E7 D0-D7 is data and E0-E7 is ECC

Syn	bits	Syn	bits	Syn	bits	Syn	bits	Syn	bits	Syn	bits	Syn	bits	Syn	bits
00	----	01	E0	02	E1	03	E0E1	04	E2	05	E0E2	06	E1E2	07	D6E5
08	E3	09	E0E3	0A	E1E3	0B	----	0C	E2E3	0D	----	0E	D7E6	0F	D5E7
10	E4	11	E0E4	12	E1E4	13	----	14	E2E4	15	D2D3	16	----	17	----
18	E3E4	19	D0E5	1A	----	1B	----	1C	----	1D	----	1E	D0D6	1F	----
20	E5	21	E0E5	22	E1E5	23	D6E2	24	E2E5	25	D6E1	26	D6E0	27	D6
28	E3E5	29	D0E4	2A	D3D4	2B	----	2C	----	2D	----	2E	----	2F	D6E3
30	E4E5	31	D0E3	32	D1E6	33	----	34	----	35	----	36	----	37	D6E4
38	D0E0	39	D0	3A	----	3B	D0E1	3C	D1D7	3D	D0E2	3E	----	3F	D2D4
40	E6	41	E0E6	42	E1E6	43	----	44	E2E6	45	----	46	D7E3	47	----
48	E3E6	49	----	4A	D7E2	4B	D0D1	4C	D7E1	4D	----	4E	D7	4F	D7E0
50	E4E6	51	----	52	D1E5	53	----	54	D4D5	55	D1D6	56	----	57	----
58	----	59	----	5A	----	5B	D4E7	5C	----	5D	----	5E	D7E4	5F	----
60	E5E6	61	----	62	D1E4	63	----	64	D2E7	65	----	66	----	67	D6E6
68	----	69	D6D7	6A	----	6B	D2D5	6C	----	6D	----	6E	D7E5	6F	----
70	D1E1	71	D3E7	72	D1	73	D1E0	74	----	75	----	76	D1E2	77	D0D7
78	----	79	D0E6	7A	D1E3	7B	----	7C	----	7D	----	7E	D3D5	7F	----
80	E7	81	E0E7	82	E1E7	83	D1D3	84	E2E7	85	----	86	----	87	D5E3
88	E3E7	89	----	8A	----	8B	D5E2	8C	----	8D	D5E1	8E	D5E0	8F	D5
90	E4E7	91	----	92	----	93	----	94	----	95	D4D7	96	D1D2	97	----
98	----	99	----	9A	----	9B	D4E6	9C	----	9D	----	9E	----	9F	D5E4
A0	E5E7	A1	----	A2	----	A3	----	A4	D2E6	A5	----	A6	----	A7	D6E7
A8	D5D6	A9	D1D4	AA	D2D7	AB	----	AC	----	AD	----	AE	----	AF	D5E5
B0	----	B1	D3E6	B2	----	B3	----	B4	----	B5	----	B6	D0D5	B7	----
B8	----	B9	D0E7	BA	----	BE	----	BC	----	BD	----	BE	----	BF	D3D7
C0	E6E7	C1	D5D7	C2	----	C3	D2D6	C4	D2E5	C5	----	C6	----	C7	----
C8	D0D3	C9	----	CA	----	CB	D4E4	CC	----	CD	----	CE	D7E7	CF	D5E6
D0	----	D1	D3D5	D2	----	D3	D4E3	D4	----	D5	----	D6	D3D6	D7	----
D8	----	D9	D4E1	DA	D4E0	DB	D4	DC	----	DD	D0D2	DE	----	DF	D4E2
E0	D2E2	E1	D3E4	E2	D0D4	E3	----	E4	D2	E5	D2E0	E6	D2E1	E7	----
E8	----	E9	----	EA	----	EB	----	EC	D2E3	ED	----	EE	----	EF	----
F0	D3E0	F1	D3	F2	D1E7	F3	D3E1	F4	D2E4	F5	D3E2	F6	----	F7	----
F8	----	F9	D3E3	FA	----	FB	D4E5	FC	D4D6	FD	D1D5	FE	----	FF	----